NEW ALGORITHMS FOR STEINER TREE ON GRAPHS

Nang-Ping Chen

Department of Electrical Engineering and Computer Sciences
and Electronics Research Laboratory
University of California, Berkeley, California 94720

ABSTRACT

The problem of Steiner Tree On Graphs is of wide applications in many fields and of special interests to global and detailed routing of IC layout. An O(|E|log|E|) time algorithm to find the three point Steiner Tree On Graphs is first developed. Two efficient heuristic algorithms for general Steiner Tree On Graphs problem are developed based on this three point method. The quality of solutions is good because they are equal or close to the optimum ones.

Introduction

Given a undirected weighted graph G=(V,E,w), where V is the set of vertices of G, E is the set of edges of G and w is the weight function which maps edges into positive real numbers. Also given a set S of original points which is a subset of V. The problem is to find a tree in G that spans S with mininmum total weight. This problem has been proved to be NP-complete[1]. Two heuristic algorithms are developed in this paper to find the sub-optimal solutions. Both have yielded good quality solutions with reasonable computational complexities.

An important application of this problem is in the global routing of IC layout, where a graph is constructed from the placement of blocks with edge weight reflecting the congestion, length and some other important factors of the routing region with which the edge is associated. Each signal is globally routed by the Steiner tree algorithm on this graph to determine the best route on the layout plane which the signal should follow.

In the following, P(A, B) represents the shortest path between A and B where A, B are vertices of G. This can always be found by the Dijkstra's algorithm. c(A, B) denotes the cost of P(A, B). Extend these notations to let A, B represent connected subgraphs of G, then P(A, B) is the shortest path that connect this two components A, B and c(A, B) is the cost of this path. P(A, B) will be empty if A and B are connected originally and in this case c(A, B) equal to zero.

Three Point Steiner Tree Algorithm

In this section we restrict ourselves to a special case of the Steiner tree problem on graphs where S contains only three points. Let S= {A, B, C}.

The usual way to find a three point sub-optimal Steiner Tree is to find the shortest path between two points first, say A and B with path P(A, B) and cost c(A, B), then find the shortest path between C and P(A, B) as shown in Figure 1. D is the point where P(C, P(A, B)) and P(A, B) meet. D may be equal to A, B or any other vertices on P(A, B). If D is not equal to A or B, then it is called a Steiner point. But this may not be the optimum solution. Let the optimum solution for the minimum cost Steiner Tree be P(A, E) +P(B, E)+P(C, E) as shown by dotted lines in Figure 1. The following equations have to be hold:

$$c(A,E)+c(B,E)\geq c(A,D)+c(B,D) \quad (1)$$

$$c(A,E)+c(B,E)+c(C,E)\leq c(A,D)+c(B,D)+c(C,D) \quad (2)$$

$$c(A,E)+c(C,E)\geq c(C,D) \quad (3)$$

$$c(B,E)+c(C,E)\geq c(C,D) \quad (4)$$

where the first equation is due to the fact that P(A, B) is the shortest path between A and B, the second one due to the fact that P(A, E)+ P(B, E)+P(C, E) is the optimum Steiner Tree, the third one is because P(C, D) is the shortest path between P(A, B) and C, otherwise P(A, E)+P(C, E) will be a shorter path between C and P(A, B) than P(C, P(A, B)) which is P(C, D) in this case and similar reason for the equation (4). Add up equations (3) and (4), we get

$$c(A,E)+c(B,E)+2*c(C,E)\geq 2*c(C,D). \quad (5)$$

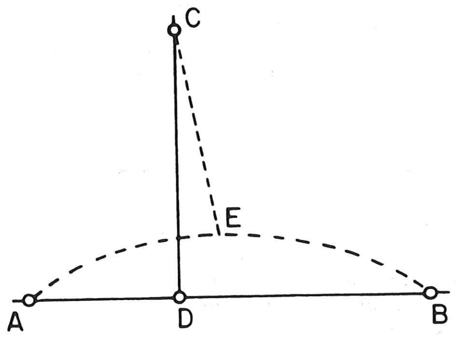


Fig. 1

Add up (5) and 2*(2), we have

$$c(A,E)+c(B,E)\leq 2*c(A,B) \tag{6}$$

The equation (6) implies that in search of the optimum Steiner Tree, the possible candidate F for Steiner point E will be within the boundary of the vertices with c(A, F)+c(B, F) smaller or equal to two times c(A, B). Also from equation (1) and (2),

$$c(C,E)\leq c(C,D) \tag{7}$$

which also gives a bound for searching E from C that the path P(C, E) has no more cost than P(C, D).

Based on the observations above the algorithm TPSTOG (Three Point Steiner Tree On Graphs) is developed as follows:

STEP 1. Do the expansion operation from A. (The expansion from a vertex or a component is to choose a vertex in V which is nearest to the starting vertex or component and has not been visited yet. Visit this vertex and mark the cost from the starting vertex or component to this vertex which is the smallest cost between this vertex and the starting vertex or component. This is the basic operation in Dijkstra's algorithm.)

STEP 2. If the expansion reaches B, then record its cost as c(A, B), else go to STEP 1.

STEP 3. Keep on doing expansion operation until cost reaches 2*c(A, B).

STEP 4. Do the expansion operation from B until cost reaches 2*c(A, B). (After this step, those vertices reached from both expansions are candidates for the Steiner point E.)

STEP 5. Do the expansion operation from C until it reaches P(A, B). Record the vertex that has minimum sum of those three expansion cost above. This is the Steiner point E. The optimum Steiner Tree can be found by tracing out P(A, E), P(B, E) and P(C, E) from the above expansions.

TPSTOG needs three expansion operations in its process. Each expansion takes time O(|E|log|E|) where |E| is the number of edges. So this algorithm has the time complexity O(|E|log|E|). To achieve this time complexity, O(d|V|) space is needed where d is the maximum vertex degree in G and |V| is the number of vertices. This space is allocated to build up a heap of edges sorted by the cost of the edge from the starting point A, B or C for the expansion operations. The space complexity is dominated by this heap and hence TPSTOG has space complexity O(d|V|).

An extension of TPSTOG can be done by observing that A, B and C need not be restricted to vertices. We can replace them by connected components of G. The problem will now become to find the smallest cost pathes that connect A, B and C into one big connected component. We can see in the following sections how this extension helps in the general Steiner Tree algorithms.

General Steiner Tree On Graphs Algorithm 1

The basic idea behind the first algorithm STOG1 (Steiner Tree On Graphs 1) is to begin with a small connected component and each time enlarge the component to include one more vertex in S in a way that tries to minimize the increasing cost due to this inclusion. The algorithm STOG1 runs as follows:

STEP 1. Find the starting component M by choosing the smallest cost three point Steiner Tree out of all possible three points in S.

STEP 2. Expanding from M to find the nearest vertex N in S to M.

STEP 3. Expanding from N to find the nearest original or Steiner point L in M to N.

STEP 4. For each path connecting L to original points or Steiner points in M do the following thing:

```
remove the path from M to
disconnect M into two
components MA, MB;
run TPSTOG on MA, MB and N;
```

STEP 5. Choose the smallest cost one in STEP 4 and M is set to this new component which contains one more original point N than the previous M.

STEP 6. If there are still some original points not included in M, then go to STEP 2.

Consider the time complexity of STOG1. Step 1 can be done with no more than |S| times TPSTOG operations by careful programming. Step 2 and 3 needs 2 expansion operations and Step 4 needs at most O(d) times TPSTOG operations where d is the maximum vertex degree of G. The loop between Step 2 and Step 6 will run at most |S|-3 times. So the worst case time complexity is O(d|S||E|log|E|). Usually STOG1 runs in a much lower time bound. The space complexity is same as TPSTOG to be O(d|V|).

General Steiner Tree On Graphs Algorithm 2

The basic idea behind STOG2 (Steiner Tree On Graphs algorithm 2) is to consider there are originally |S| components each containing only one original point in the beginning. Components are combining with each other into a bigger one in the process until there is only one component left and hence it is the solution we want. STOG2 runs as follows:

STEP 1. Set up |S| components by letting each to contain one original point only.

STEP 2. Find the smallest cost Steiner Tree that connect three components out of all possible components.

STEP 3. Combine these three components into a big component.

STEP 4. If there are more than three components exists, then go to STEP 2, else if there are only two components left, then do an expansion operation to find the shortest path between them and combine them into one component.

Step 2 will take no more than k times TPSTOG where k is the current number of components. The loop between Step 2 and 4 will be executed no more than |S|/2 times. So there are at most $O(|S|^2)$ times that TPSTOG should run in the process. The time complexity is therefore $O(|S|^2|E|\log|E|)$ and the space complexity is still O(d|V|).

Experiments

The algorithms are coded in C language on VAX 11/780 with UNIX operating system. The results are

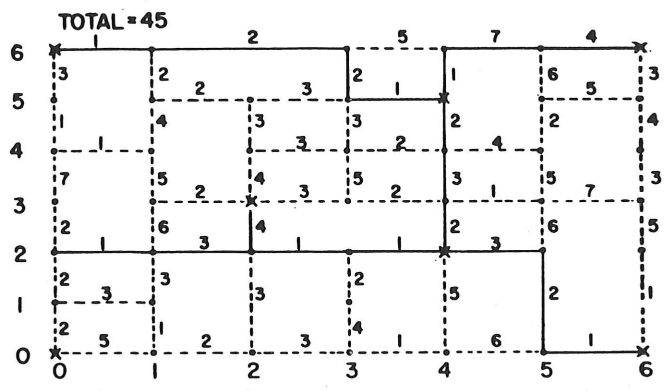


Fig. 2

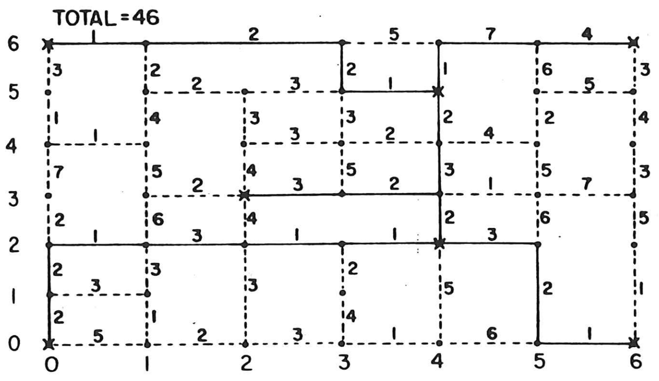


Fig. 3

encouraging because they always get the solutions which are equal or close to the optimum ones. The result can be even better if we run both algorithms for the same problem and choose the better one. An example for STOG1 is shown in Figure 2 where the background dotted line is the graph G with cost marked at each edge, original points are marked with X, the solid line is the Steiner tree found and the cost is shown on the upper left corner. The same example for STOG2 is shown in Figure 3.

Conclusion

One three point Steiner Tree on Graphs algorithm and two heuristic algorithms for general Steiner Tree on Graphs are proposed in this paper. The result is satisfactory in both solution quality and computational complexity. These algorithms are used in the global router of the Building Block Layout System developed at UC Berkeley and successful results are observed.

Acknowledgement

The author wishes to thank to Prof. E.S. Kuh for his support and guidance of this project. The programming help from Howard Chen is gratefully acknowledged.

Research sponsored by the National Science Foundation grant ECS-8201580 and the Air Force Office of Scientific Research (AFSC) United States Air Force contract F49620-79-c -0178.

References

1. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of computer computations (R.E.Miller. J.W.Thatcher eds), pp. 85-104. New York. Plenum Press 1972.

2. Lee, J.H.: Aspects of Layout of Multi-Net in Multi-Layer over Rectilinear Metric, Ph.D. dissertation, University of Pittsburgh, Pittsburgh, 1975.

3. Gilbert, E.N. and Pollak, H.O.: Steiner Minimal Tree, SIAM J. Appl. Math., Vol. 16, no. 1, pp. 1-29, Jan., 1968.

4. Moore, E.F.: Shortest Path Through a Maze, Proc. of Intl. Sym. on Switching Circuits, pp. 285-292 Harvard Univ. Pree, 1957.

5. Hakimi, S.L.: Steiner's Problem in Graphs and Its Implications , Networks, Vol. 1, pp. 113-133, 1971.

6. Kou, L, Markowsky, G. and Berman, L.: A Fast Algorithm for Steiner Trees, Acta Informatica 15, pp. 141-145, 1981.

7. Dijkstra, E.N.: A Note On Two Problems in Connection with Graphs, Numer. Math. 1. pp. 269-271, 1959.

8. Hwang, F.K.: On Steiner Minimal Trees with Rectilinear Distances, SIAM J. Appl. Math. 30. pp. 104-114, 1976.

9. Hanan, M.: On Steiner's Problem with Rectilinear Metric, SIAM. J. Appl. Math. Vol. 16, no. 1, pp.255-265, 1966.

10. Yang, Y.Y. and Wing, O.: Sub-optimal Algorithm for a Wire Routing Problem, IEEE Tran. Circuit Theory, Vol. CT-19, pp. 508-511, Sep., 1972.

ON GRAPH THEORETIC MODELS FOR THE CIRCUIT LAYOUT PROBLEM

Kazuo Nakajima and Moses Sun

Department of Electrical Engineering/Computer Science
Texas Tech University
Lubbock, TX 79409

ABSTRACT

We consider the circuit layout problem on integrated circuits and single layer printed circuit boards. In the past a cycle has been proven to be a powerful model for use in a topological approach to solving the problem. In this paper we show that a star is as powerful as a cycle for that purpose. The use of this graph model is further supported by a recently developed efficient algorith for testing the planarity of a partially oriented graph representing the circuit.

I. INTRODUCTION

We consider the circuit layout problem on integrated circuits and single layer printed circuit boards. A circuit consists of a set of components, each of which is in turn a set of terminals, and a set of electrical interconnections or "nets" among two or more terminals. Each net specifies a set of terminals to be interconnected with a single conducting path or "wire". The problem is to position the components and wires in a plane without crossover of the net wiring.

The topological aspect of this problem is usually answered by testing the planarity of a graph representing the circuit. The main problem is to realize certain constraints imposed upon the circuit in the construction of such a graph. Two basic constraints are:

C1) The terminals of a component appear on its physical boundary in a specified order.

C2) The external connections of the circuit appear on the outside boundary in a prescribed order.

Although a limited amount of "under-component wiring" is allowed in certain situations, we do not consider the wiring problem of this type here.

Various graph models have been proposed [1-7]. A wheel [2] has been proven useful in the realization of the above constraints; however, this model may lead to a situation in which the corresponding component is placed on board with its back side up. Although a cycle is a more primitive graph model than a wheel, it has recently shown [6,7] that the model with additional constraints overcomes this difficulty. In [6], vanCleemput introduced an orientation of each vertex on a cycle in order to specify the sequence in which its incident edges appear around the vertex. On the other hand, Masuda et al. [7] introduced a direction of each edge on a cycle and imposed that no vertex be placed inside the cycle. Their model for the entire circuit becomes a mixed graph, which they called an L-graph. They developed a linear time algorithm for testing the planarity of an L-graph. We should note that both approaches lead to a special case of a partially oriented graph introduced by vanCleemput [6].

A star has also been proposed as a graph model to realize the constraints. This was originally introduced by Goldstein and Schweikert [3] to represent a net. Uyehara et al. [5] later used this model to realize constraint C1 by adding an orientation to the center vertex of a star. Their graph model for the entire circuit becomes a general partially oriented graph. They proposed a polynomial time algorithm for testing the planarity of such a graph. The authors [8,9] have recently obtained a linear time implementation of their algorithm.

In [6] van Cleemput not only considered the above basic constraints, but also showed that appropriate modifications of a cycle can realize several additional properties of physical circuits such as physical and logical equivalences of terminals of a component and logical equivalence of subcomponents of a component.

In this paper, we show that with appropriate modifications a star can realize the same additional properties of physical circuits as a cycle can. Furthermore, using our recent algorithm [8,9], the planarity testing problem is solvable in linear time for a circuit containing these properties and the above constraints. We should note, however, that the Masuda et al. [7] algorithm can be applied to the graph theoretic model of vanCleemput [6] if necessary modifications are made.

II. GRAPH MODELS FOR THE BASIC CONSTRAINTS

In this section, we present the graph models for the two basic constraints C1 and C2 mentioned previously. Basically, both a multi-terminal component and the outside boundary of a circuit having external connection terminals are modeled by a star with an appropriate orientation of its center vertex. The resultant graphs are partially oriented [6].

2.1. Definitions

Let G(V,E) be a finite undirected graph with a vertex set V and an edge set E. For a vertex $v \epsilon V$, let $A(v) = \{u|(v,u) \epsilon E\}$. A cyclic permutation of the vertices in A(v) is called an orientation o(v) of a vertex $v \epsilon V$. An orientation O(G) of a graph G is a mapping of the vertex set V into the set of orientations of all vertices in V. A graph G(V,E) is said to be partially oriented if O(G) is defined for a proper subset of V. A vertex for which an orientation is defined is called an oriented vertex.

A graph G is said to be embeddable in a plane if there exists a mapping of the vertices and edges of the graph into the plane such that (1) each vertex is mapped into a distinct point, (2) each edge (u,v) is mapped onto a continuous line connecting the two points onto which the vertices u and v are mapped, and (3) no two lines which are the mappings of distinct edges, share any points, except in their common end-points. A partially oriented graph G is called planar if it can be embedded in the plane such that for each oriented vertex v, a clockwise sweep around its mapped point encounters those lines that are the mappings of the edges incident on the vertex v, in the order prescribed by the orientation o(v).

2.2. Modeling a Circuit by Oriented Stars

Consider the circuit shown in Fig. 1. It consists of three 4-terminal components C_1, C_2, C_3, three external connection terminals T_1, T_2, T_3 located on the outside boundary B, and seven nets $N_1,\ldots, N_7$. Figure 2 shows a partially oriented graph representing the circuit. Each component is modeled by a star with the orientation of its center vertex being specified by the clockwise sequence of the outer vertices corresponding to the four terminals. The boundary B is also modeled by a star; however, its center vertex is oriented in the counter clockwise sequence of the terminals T_1, T_2, and T_3. Finally, each net is represented by a star without orientation. Note that if a net consists of two terminals, it is modeled by a single edge rather than a star with two outer vertices.

It is clear that the circuit is laid out on a plane without wire crossings if and only if its associated partially oriented graph is planar.

III. GRAPH MODELS FOR ADDITIONAL PROPERTIES

In the preceding section, we modeled a cyclic ordering of the terminals of a component by the orientation of the center vertex of a star. In this section, we consider other relations between the terminals of a component; namely, physical and logical equivalences of terminals, and subcomponent equivalence. In vanCleemput's paper [6] these properties of physical circuits are realized by modifying the original cycles corresponding to the components. We show that these properties are also realizable by stars with appropriate modifications.

3.1. Physical Equivalence of Terminals

Terminals of a component are said to be physically equivalent if they are equipotential. This means that a net can be connected to any one of these terminals.

Consider the component with ten terminals shown in Fig. 3(a), where terminals 2, 9 and 4, 7 are physically equivalent, respectively. Then, this component with these physical equivalences is modeled by the partially oriented graph shown in Fig. 3(b).

3.2. Logical Equivalence of Terminals

Terminals of a component are said to be logically equivalent if they have identical logical functions. This means that these terminals are interchangeable in order to obtain a better layout.

Consider the 10-terminal component shown in Fig. 4(a), where terminals 2, 3, 4 and 5, and 8, 9 and 10 are logically equivalent, respectively. Then, this component with these logical equivalences is modeled by the partially oriented graph shown in Fig. 4(b). We should note that this model is appropriate for logical equivalence if all logically equivalent terminals are physically adjacent.

3.3. Logical Equivalence of Subcomponents

Subcomponents of a component are said to be logically equivalent if they are physically identical and logically interchangeable. This means that a component should be assigned to a particular group of nets to be connected to one of its equivalent subcomponents in function of an optimal layout.

Consider the component with twelve terminals shown in Fig. 5(a), where the three set of terminals {1,2,3}, {4,5,6}, and {7,8,9} are logically equivalent. Then, this component having these logically equivalent subcomponents is modeled by the partially oriented graph shown in Fig. 5(b).

We should note that each of logically equivalent subcomponents must be made up of physically adjacent terminals which appear on the component's boundary in the same order. Furthermore, in our example, the three subcomponents are physically adjacent. When not all equivalent subcomponents are physically adjacent, it is not always possible to obtain a partially oriented graph model whose embedding in a plane always leads to a feasible layout of a circuit. This problem is not resolved in the vanCleemput approach [6], either.

IV. CONCLUSIONS

We have proposed the graph theoretic models for the circuit layout problem. We have shown that multiterminal components and the outside boundary of a circuit can be realized by stars with orientation. We have then shown that physical and logical equivalences of terminals of a component and logical equivalence of subcomponents of a component can be modeled by appropriately modifying oriented stars. The same properties of a circuit were previously modeled by cycles with proper modifications [6].

In both approaches, the resultant graph models for the entire circuit are partially oriented graphs. However, our graph model requires fewer vertices than the vanCleemput model [6]. Using our efficient algorithm recently developed [8,9], the problem of testing the planarity of the resultant partially oriented graphs is solvable in linear time. Finally, we should note that the Masuda et al. algorithm [7] can be applied to the vanCleemput graph model [6] if appropriate modifications are made.

REFERENCES

[1] O. Wing, "On Drawing a Planar Graph," IEEE Trans. on Circuit Theory, CT-13, 1966, pp. 112-114.

[2] W. L. Engl and D. A. Mlynski, "Embedding a Graph in a Plane with Certain Constraints," IEEE Trans. on Circuit Theory, CT-17, 1970, pp. 250-252.

[3] A. J. Goldstein and D. G. Schweikert, "A Proper Model for Testing the Planarity of Electrical Circuits," Bell System Tech. J. 52, 1973, pp. 135-142.

[4] M. C. Vanlier and R. H. Otten, "On the Mathematical Formulation of the Wiring Problem" Int. J. of Circuit Theory and Applications, 1, 1973, pp. 137-147.

[5] T. Uyehara, H. Shiraishi, O. Takahashi, and T. Kojima, "Embedding a Graph in a Plane with Local Constraints," Proc. IEEE Int. Symp. on Circuits and Systems, San Francisco, 1974, pp. 22-25.

[6] W. M. vanCleemput, "Mathematical Models for the Circuit Layout Problem," IEEE Trans. on Circuits and Systems, CAS-23, 1976, pp. 759-769.

[7] S. Masuda, T. Kashiwabara, and T. Fujisawa, "A Layout Problem on Single Layer Printed Circuit Board," IECE of Japan, Tech. Rep. CAS81-19, pp. 93-100.

[8] K. Nakajima and M. Sun, "On an Efficient Implementation of a Planarity Testing Algorithm for a Graph with Local Constraints," Proc. 20th Annual Allerton Conference on Communication, Control, and Computing, Univ. of Illinois, Urbana, IL, Oct. 1982, 656-664.

[9] K. Nakajima and M. Sun, "Efficient Planarity Testing for Partially Oriented Graphs," in preparation.

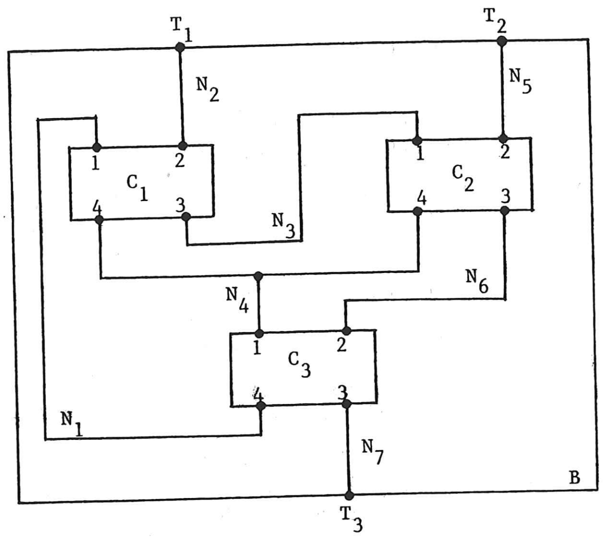
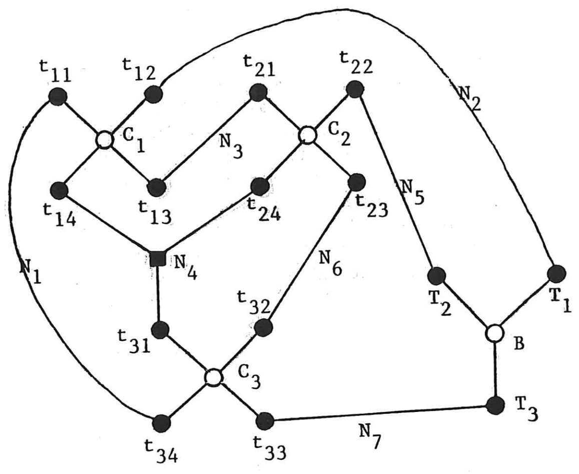


Fig. 1. Sample Circuit.



○ oriented vertex

● normal vertex

■ net vertex

Fig. 2. Partially oriented graph modeling the circuit in Fig. 1.

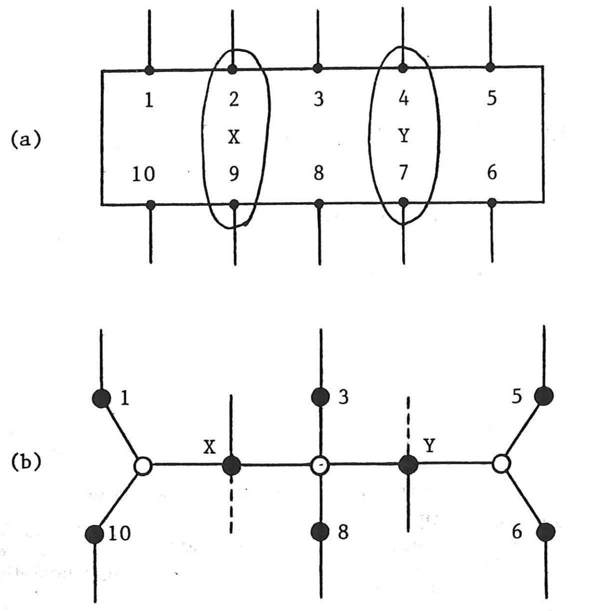


Fig. 3. (a) Component with physical equivalence. (b) Partially oriented graph modeling physical equivalence.

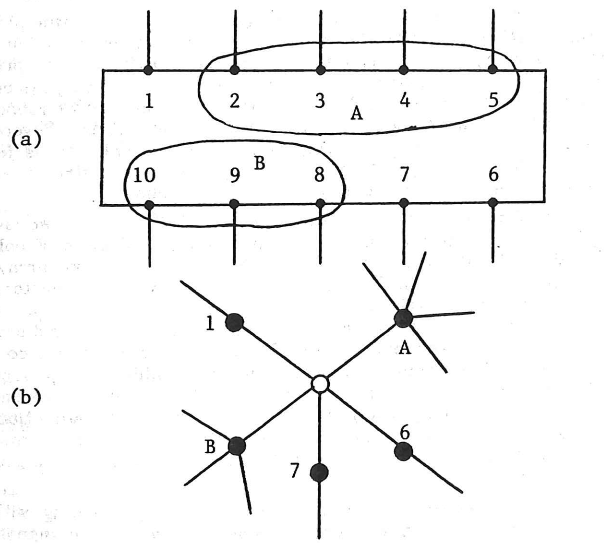


Fig. 4. (a) Component with logical equivalence. (b) Partially oriented graph modeling logical equivalence.

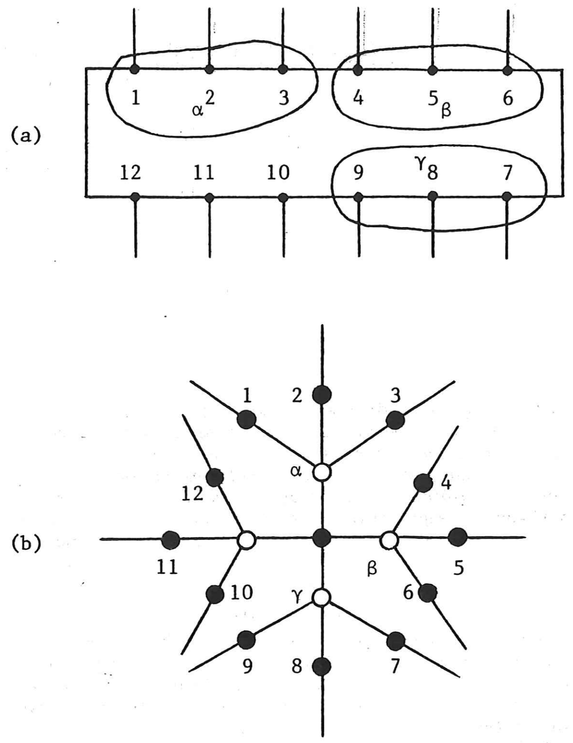


Fig. 5. (a) Component with subcomponent equivalence. (b) Partially oriented graph modeling subcomponent equivalence.